Reversible Data Hiding Technique using AMBTC based Bitmap Manipulation

P. Uma
Research Scholar
Department of Computer Science
Mother Teresa Woman's University, Kodaikanal

S. Vimala
Associate Professor
Department of Computer Science
Mother Teresa Woman's University, Kodaikanal

ABSTRACT
In this research work, it has been proposed to use images to hide the secret data as it is difficult to extract secret data from the image. Using images to hide and transmit secret data, leads to additional overhead of increased cost associated with storage and transmission cost, as the image requires more space for storage. However, hiding data inevitably destroys the host image, even though the distortion is imperceptible. To overcome such drawbacks, image compression techniques are used for both, hiding the secret data and to reduce the storage and transmission cost. To enhance the hiding capacity and maintain the quality of the host image after embedding hidden data, we present a high payload reversible data hiding scheme that is based on the Absolute Moment Block Truncation Coding (AMBTC) compression domain. We exploit the feature of inter block redundancy in an AMBTC compressed image to improve the coding efficiency of compressed images. Normally, the AMBTC technique transforms an input image into a set of blocks and each block in turn is transformed into a set of a bit-plane and two quantizers. But in the proposed method, these blocks are categorized into Shade and Edge blocks based on the magnitude of amplitude values. For a Shade block, two values are stored, one being the block mean and the other value being the secret data. Two bytes of secret data is stored creating an illusion that the bitplane is stored. For Edge blocks, high mean and low mean are stored as two quantizers and a respective bitplane is generated. The Shade block helps in embedding 3 bytes of secret data leading to higher embedding capacity and increases the complexity of identifying the existence of secret data. It also compresses the stego-image without much degradation in the visibility of stego image.

Keywords
AMBTC Compression, Embedding Capacity, Reversible Data Hiding, Stego image.

1. INTRODUCTION
Reversible data hiding (RDH) is a technique which embeds secret data into a cover image and can extract the embedded data and recover the original image with lossless. Cover image refers to the image used for carrying the embedded data. Embedded data is known as payload and the image with embedded data is called stego image [1]. Reversible data hiding technique is used in some special applications such as Military images, Medical Images and Forensics, where the exact recovery of the original cover image is inevitable. Data hiding techniques can be carried out in three domains [2] namely; Spatial Domain [3, 4], Frequency Domain [5, 6, 7], and Compression Domain [8, 9, 10]. In the Spatial Domain based data hiding techniques, the secret data is embedded by modifying the pixel values directly. These techniques are the simplest and have the least computational complexity. In most of the spatial data hiding techniques, least significant bits (LSB) [11] and pixel value differences (PVD) [12] are exploited for embedding the secret data. In frequency domain based data hiding techniques, the original image is first transformed into frequency coefficients and then the secret data is embedded as part of transformed frequency coefficients. In compression domain based data hiding techniques, the original image is first compressed into compressed codes using any available image compression methods such as Joint Photographic Experts Group (JPEG) [13], Vector Quantization (VQ) [8], Graphics Interchange Format (GIF) [14], and Block Truncation Coding (BTC) [15] and then the secret data is embedded as part of the compressed codes. The secret data that is hidden as part of the compressed image is not easily visible and hackers or attackers will not usually be wary on stego-images.

Over the last few years, many hiding schemes have been proposed based on Block Truncation Coding (BTC) [16–22], which has been the most efficient, simplest and fastest compression method. In 2008, Chang et al. proposed a novel reversible data hiding method using BTC that compresses color images. It requires three bitmaps and three pairs of quantization levels for reconstruction. Genetic algorithm (GA) is applied to find an approximate optimal common bitmap to replace the original three bitmaps to improve the coding efficiency. The secret data is embedded in the common bitmap and the quantization levels of each block use the properties of side matching and the order of quantization levels to achieve reversibility [9]. In 2010, Chen et al. proposed an improved data hiding scheme to embed the secret data in compressed bit streams and the quality of the image is also maintained even after embedding [16]. The difference of quantization levels for each block is determined whether the only 1 bit of secret data is to be hidden or to toggle bits in the bitmap to hide more bits. This scheme is very simple and does not require complex computations. In 2011, Li et al. presented the histogram shifting and bitplane flipping technique based on BTC compressed images to improve the hiding capacity and maintaining image quality [17]. In 2013, Sun et al. presented a novel BTC-based reversible hiding scheme by adopting a joint neighbor coding technique to embed the secret data into quantization levels [18]. In 2015, Lin et al. presented a high payload reversible data hiding scheme based on Absolute Moment Block Truncation Coding (AMBTC). It discovers the redundancy in a block of AMBTC compressed images to determine if a block is embeddable or non-embeddable. In this scheme, four disjoint sets are created for embeddable blocks to embed secret data using different combinations of the mean and standard deviation [21]. This scheme adopts both spatial and compression domain because this method utilizes the concept of AMBTC and they do not compress the image so that the stego image is not the BTC.
and bitmap. In edge blocks, embedding capacity is also increased without any distortion.

2.2.1 Embedding Phase
Input: Input image I of size M x N, Secret data sequence S and Threshold value (Th)
Output: A stego AMBTC compressed code (I_s).
Step1: Divide the input image block I into non-overlapping 4 x 4 blocks.
Step 2: For each block, calculate two quantization values and bitmap are generated using AMBTC method.
Step 3: Compute the absolute difference (d_i) between the two quantization values (a_i, b_i) such that, 
\[ d_i = |a_i - b_i| \]
Step 4: if \( d_i > Th \), the block is edge block, fetch the one bit s1 from S. If s1 is equal to 1, the bitmap Bi is inversed and two quantization levels are exchanged. Such that original compressed code \( (a_i, b_i, B_i) \) are transformed into \( (b_i, a_i, B_i) \) and then add \( (b_i, a_i, B_i) \) into \( I_s \).
Step 5: if \( d_i \leq Th \), the block is shade block. Fetch the 16 bits s2 from S. The bitmap is replaced with s2 and form the new bitmap \( B_i \) and recalculate the two quantization levels \( (a_i, b_i) \) based on new bitmap \( B_i \).
Step 6: If \( |a_i - b_i| \leq Th \), recalculate quantization levels are maintained the smoothness of the block and then add \( (a_i, b_i, B_i) \) into \( I_s \). Otherwise \( |a_i - b_i| > Th \), two old quantization levels \( (a_i, b_i) \) maintain the smoothness of the block and then add \( (a_i, b_i, B_i) \) into \( I_s \).
Step 7: Repeat steps 2 to 6 until all the image blocks are processed.

2.2.2 Extracting Phase
Input: A Stego compressed codes \( I_s \) size of \( \frac{M}{4} \times \frac{N}{4} \) trios \( (a_i, b_i, B_i) \) where \( i = 1, 2, ..., \frac{M}{4} \times \frac{N}{4} \) and threshold Th.
Output: Secret bit sequences (SE)
Step 1: For each trio \( (a_i, b_i, B_i) \) in \( I_s \), compute absolute difference \( d_i \), such that \( d_i = |a_i - b_i| \).
Step 2: if \( d_i > Th \), one bit s1 is extracted. if \( a_i > b_i \), s1 can be 1, otherwise s1 can be 0 and then add extracted bit s1.

International Journal of Computer Applications (0975 – 8887)  
Volume 176 – No. 12, April 2020
3. PROPOSED METHOD
The proposed method consists of three stages: i. AMBTC based compression, ii. Data Hiding and iii. Data Recovery.

3.1 The AMBTC Compression Stage
The input image is compressed using AMBTC technique. Given an M x N sized gray scale original image X, the following are the steps followed to transform the image into a compressed form based on AMBTC.

Step 1: The original gray scale image of size M x N is divided into non-overlapping blocks of size 4 x 4.
Step 2: Calculate the mean of the block using the equation (1).
Step 3: Compute the HMean and LMean values using (2) and (3).
Step5: Set the threshold (TH).
Step6: Find the difference D between HMean and LMean.
Step7: If \( D \leq TH \), the block is Shade block.
Step8: If \( D > TH \), the block is Edge block. Generate the Bitmap (BM) using (4) and compute the HMean and LMean values.
Step9: Repeat the steps 3 to 8 for each block.

3.2 The Data Hiding Stage
The secret data is hidden as part of the compressed image. The proposed method is a reversible data hiding algorithm. Before embedding, the secret data bit sequence \( S = \{s_1, s_2, \ldots \} \) undergoes permutation in order to enhance the security by increasing the identification complexity. In AMBTC, all input blocks are treated equally and for each block, a set of two quantizers and a Bitmap (BM) of size 16 bits are generated. In the proposed method, an indicator bit (1/0) is used to represent a shade block (0) and an edge block (1).

Each compressed block is transformed into a bit sequence of size 49 bits as follows:

\[
[\text{IndicatorBit} \| Q1 \| Q2 \| BP] \]

For a shade block, two values are stored. The first value is the Mean of the block (Q1) and the second value is the 8-bit secret data (Q2). As a Bitmap is not necessary for a shade block, a 32 bit secret data is stored as Bitmap (BM) which increases the embedding capacity. Hence in a shade block, a secret data of 40 bits is stored out of 49 bits.

For an edge block, HMean (Q1) and LMean (Q2) and a 32-bit Bitmap are generated. HMean and LMean are block-specific. In the Bitmap, out of 2 bits, the first bit indicates the nature of the input image’s pixel value and the second bit represents the secret data. In edge blocks, 16 bits of secret data is stored out of 49 bits.

The formation of BM for an edge block is as follows:

\[
\begin{bmatrix}
    b_1 & b_2 & b_3 & b_4 \\
    b_5 & b_6 & b_7 & b_8 \\
    b_9 & b_{10} & b_{11} & b_{12} \\
    b_{13} & b_{14} & b_{15} & b_{16}
\end{bmatrix}
\]

3.3 Second Level of Compression
To reduce the Bit rate, LMean value is subtracted from HMean and the difference is stored as LMean, it requires only less number of bits. This leads to further reduction in image data. By subtracting the difference from HMean, LMean can be recovered later while reconstructing the image from compressed data.

For example, HMean = 236, LMean = 225, \( D = 236 - 225 = 11 \). Generally, we need 8 bits to store HMean and LMean values. But to store the difference in this case, we need only 4 bits.

Embedding of secret data as part of Shade block is explained in Fig. 1.

![Fig. 1. Example of Secret data embedding in Shade Block](image)

Embedding of secret data as part of Shade block is explained in Figure 1 and Embedding of secret data as part of Edge Block is explained in Figure 2.

![Fig. 2. Example of Secret data embedding in Edge Block](image)

3.4 Data Extraction and Recovery Stage
In case of a Shade block, the second quantizer (Q2) and the Bitplane (BP) give the secret data. The compressed image block is reconstructed by filling all pixel values with the Mean value (Q1). In Edge block, 16 bits of secret data is retrieved
by extracting the second bit from each element of Bitplane.

The current block is replaced with HMean and LMean based on the original Bitplane using (5).

\[
RB(i, j) = \begin{cases} 
LMean & \text{if } BP(i, j) == 0 \\
HMean & \text{if } BP(i, j) == 1 
\end{cases} 
\tag{5}
\]

The flow of Data embedding stage is explained in Fig. 3.

![Flowchart of Data Embedding Stage](image)

**Fig. 3. The flowchart of Data Embedding Stage**

For example, in shade block,

\[
OutputCS = \{00110100001110101110110100010110\}
\]

In the above output code stream, indicator bit is 0, it is a shade block. We retrieve Q1 as Mean and the value is 104. Q2 and 32-bit plane are secret data.

\[
SecretData = \{10110001000111011011011101\}
\]

\[
RB = \begin{bmatrix}
104 & 104 & 104 & 104 \\
104 & 104 & 104 & 104 \\
104 & 104 & 104 & 104 \\
104 & 104 & 104 & 104 \\
\end{bmatrix}
\]

In Edge block,

\[
OutputCS = \{10111010010001110110110100010111\}
\]

Indicator bit is 1, it is an edge block. We retrieve Q1 and Q2 as HMean and D respectively.

HMean=117, D=77, LMean=117-77=40.

\[
EmbeddedBP = \begin{bmatrix}
01 & 01 & 11 & 10 \\
01 & 00 & 11 & 10 \\
00 & 00 & 10 & 11 \\
01 & 00 & 10 & 11 \\
\end{bmatrix}
\]

\[
SecretData = \begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[
BP = \begin{bmatrix}
00 & 01 & 11 & 10 \\
00 & 11 & 10 & 11 \\
11 & 10 & 10 & 11 \\
00 & 01 & 11 & 10 \\
\end{bmatrix}
\]

\[
RB = \begin{bmatrix}
40 & 40 & 117 & 117 \\
40 & 40 & 117 & 117 \\
40 & 40 & 117 & 117 \\
40 & 40 & 117 & 117 \\
\end{bmatrix}
\]

The workflow of data extraction and recovery stage is explained in Figure 4.

![Flowchart of Data Extraction and Recovery Stage](image)

**Fig 4. The workflow of Data Extraction and Recovery Stage**

4. RESULTS AND DISCUSSIONS

All experiments are performed with the benchmark gray scale images of size 512 x 512 pixels. The input images are Airplane, Lena, Boats, Bridge, Man, Peppers, Sailboat, Zelda, Girl and Couple. Fig. 5 shows the input images taken for the study. The algorithms are implemented using MATLAB R2014b. The Hardware used is, the Intel® Pentium® 1.90 GHZ processor with 8GB RAM.
In our proposed method, the secret data is embedded into compressed image and the image is called Stego Image. Then we extract the embedded secret data from Stego Image thereby leading to original Cover Image. The Peak Signal to Noise Ratio (PSNR) is used to measure the quality of the compressed image using (6).

\[
PSNR = 10 \log_{10} \left( \frac{255^2}{MSE} \right) \text{dB} \tag{6}
\]

where, MSE is the Mean Square Error which is used to measure the error between the original image and compressed image and is computed using (7).

\[
MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (I(i, j) - \hat{I}(i, j))^2 \tag{7}
\]

where, M is the number of rows and N is number of columns. I(i, j) and \( \hat{I}(i, j) \) denote the Original image and Compressed image respectively.

The bitrate is used to analyze the coding efficiency of any Compression algorithm. The bitrate is computed using the equation (8), where CS represents the length of compression code, M \( \times \) N is the size of the original image. The bitrate measured is represented in terms of bits per pixel (bpp).

\[
\text{bitrate} = \frac{CS}{M \times N} \text{(bpp)} \tag{8}
\]

To illustrate the performance of our proposed method, the results of our scheme with various threshold values ranging between 5 to 25 are shown in Table I. Input images of size 512 \( \times \) 512 are taken for the study and the optimal block size is taken as 4 \( \times \) 4 pixels. The proposed method is tested with five different images and the average embedding capacities obtained are 366589, 452393, 507274, 548740, 571092 and the average PSNR and BPP of the images are 39.64, 38.24, 37.48,36.90, 36.48 and 2.87, 2.94, 2.99, 3.02, 3.03 respectively.

The performance of the proposed scheme is compared with the existing schemes (reversible schemes) proposed by Chang et al.[9], Li et al.[17], Sun et al.[18], Lin et al.[21] and Chan et al.[22] in terms of embedding efficiency (EE) and is shown in Table II. The Embedding Efficiency (EE) is computed using the equation (15).

\[
EE = \frac{\text{Embedding Capacity}}{\|CS\|} \times 100 \tag{15}
\]

where \( \|CS\| \) is the size of the compressed image in terms of bits.

Table I. Comparison of PSNR, Embedding Capacity and BPP of Proposed method with Various Thresholds

<table>
<thead>
<tr>
<th>Images</th>
<th>Parameters</th>
<th>THRESHOLD Range (5-25)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TH=5</td>
<td>TH=10</td>
</tr>
<tr>
<td>Lena</td>
<td>PSNR</td>
<td>37.58</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
<td>417688</td>
</tr>
<tr>
<td></td>
<td>BPP</td>
<td>2.90</td>
</tr>
<tr>
<td>F16</td>
<td>PSNR</td>
<td>37.33</td>
</tr>
<tr>
<td></td>
<td>Capacity</td>
<td>462544</td>
</tr>
<tr>
<td>Image</td>
<td>BPP</td>
<td>PSNR</td>
</tr>
<tr>
<td>-------------</td>
<td>-----</td>
<td>------</td>
</tr>
<tr>
<td>Sail Boat</td>
<td>2.96</td>
<td>3.00</td>
</tr>
<tr>
<td>Girl</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Toys</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Barbara</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Baby</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Baboon</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Boats</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Bridge</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Goldhill</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Girface</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Pepper</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Tiffany</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Man</td>
<td>3.01</td>
<td>3.02</td>
</tr>
<tr>
<td>Zelda</td>
<td>3.01</td>
<td>3.02</td>
</tr>
</tbody>
</table>
Table II. Comparison EE (%) for the proposed scheme and Existing Schemes

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Parameters</th>
<th>Chang et al. [9]</th>
<th>Li et al. [17]</th>
<th>Sun et al. [18]</th>
<th>Lin et al. [21]</th>
<th>Lin et al. [22]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>Capacity</td>
<td>31011</td>
<td>16789</td>
<td>64008</td>
<td>76671</td>
<td>262112</td>
<td>417688</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>480096</td>
<td>2097152</td>
<td>759860</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>6%</td>
<td>3%</td>
<td>12%</td>
<td>16%</td>
<td>13%</td>
<td>55%</td>
</tr>
<tr>
<td>F16</td>
<td>Capacity</td>
<td>30518</td>
<td>17659</td>
<td>64008</td>
<td>80009</td>
<td>261984</td>
<td>462544</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>474264</td>
<td>2097152</td>
<td>775529</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>6%</td>
<td>3%</td>
<td>12%</td>
<td>17%</td>
<td>12%</td>
<td>60%</td>
</tr>
<tr>
<td>SailBoat</td>
<td>Capacity</td>
<td>28766</td>
<td>17082</td>
<td>64008</td>
<td>84791</td>
<td>262096</td>
<td>334384</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>495432</td>
<td>2097152</td>
<td>747747</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>5%</td>
<td>3%</td>
<td>12%</td>
<td>17%</td>
<td>13%</td>
<td>45%</td>
</tr>
<tr>
<td>Girls</td>
<td>Capacity</td>
<td>30962</td>
<td>16990</td>
<td>64008</td>
<td>108217</td>
<td>262128</td>
<td>379552</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>544104</td>
<td>2097152</td>
<td>758928</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>6%</td>
<td>3%</td>
<td>12%</td>
<td>20%</td>
<td>13%</td>
<td>50%</td>
</tr>
<tr>
<td>Toys</td>
<td>Capacity</td>
<td>27870</td>
<td>17761</td>
<td>64008</td>
<td>70889</td>
<td>262112</td>
<td>487120</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>455112</td>
<td>2097152</td>
<td>774345</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>5%</td>
<td>3%</td>
<td>12%</td>
<td>16%</td>
<td>13%</td>
<td>63%</td>
</tr>
<tr>
<td>Barbara</td>
<td>Capacity</td>
<td>30151</td>
<td>16755</td>
<td>64008</td>
<td>87264</td>
<td>262128</td>
<td>262216</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>510048</td>
<td>2097152</td>
<td>700832</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>5%</td>
<td>3%</td>
<td>12%</td>
<td>17%</td>
<td>13%</td>
<td>37%</td>
</tr>
<tr>
<td>Average</td>
<td>Capacity</td>
<td>29880</td>
<td>17173</td>
<td>64008</td>
<td>84640</td>
<td>262093</td>
<td>390584</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>524288</td>
<td>524288</td>
<td>524288</td>
<td>493176</td>
<td>2097152</td>
<td>752874</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>6%</td>
<td>3%</td>
<td>12%</td>
<td>17%</td>
<td>13%</td>
<td>52%</td>
</tr>
</tbody>
</table>

It is observed from Table II, the results in terms of Embedding Capacity and Embedding Efficiency are better with proposed method when compared to that of the existing similar methods and are highlighted. An average embedding capacity of 390584 bits and embedding efficiency of 52% are obtained with the proposed method.

5. CONCLUSION

A novel reversible data hiding method using Absolute Moment Block Truncation Coding based on Edge and Shade block approach is presented in this paper. Hence adaptive embedding is possible based on the type (Shade or Edge) of the input image block. Our proposed method is suitable for hiding large volume of information in multimedia. The maximum embedding capacity obtained with the existing method [22] is 1,28,491 bits less than that of proposed, which is a significant improvement. Similarly, the maximum average Embedding Efficiency obtained with the existing method [21] is 35% less than that of the proposed method. The proposed method, hence outperforms than all the existing similar methods mentioned in this both in terms of Embedding Capacity and Embedding Efficiency. Instead of using 16 bits out of 32 bits of Bitplane of an Edge block for embedding the secret data, the embedding capacity can further be improved by trying to make use of all 32 bits in the future without loss of quality in the Stego Image.

6. REFERENCES


54


